Trabalho 1 - Algoritmos e Incerteza
Entrega: 14 de Outubro

Marco Molinaro

Teoria

1 Online Load Balancing

Considere o problema **Load Balancing** no modelo online. Existem m máquinas e n jobs que tem que ser processados em uma das maquinas. O tempo de processamento dos jobs depende da maquina atribuída; usamos $p_i^t \geq 0$ para denotar o tempo que o job t leva caso atribuído na maquina i. A *carga de uma máquina* é a soma dos tempos de processamento os jobs atribuidos a ela.

Jobs chegam um a um, ou seja, recebemos a sequencia de vetores $p^1, p^2, \ldots, p^n$ item a item, e quando um job chega ele tem que ser atribuído a uma das máquinas (somente com o conhecimento dos jobs vistos ate agora). O objetivo é minimizar a maior carga das máquinas (i.e. queremos que todas as máquinas tenham carga "pequena") ao fim desse processo.

Ou seja, escolhe online vetores $x^1, \ldots, x^n \in \{0,1\}^m$ satisfazendo $\sum_i x_i^t = 1$ (então $x_i^t = 1$ é equivalente a atribuir job t à máquina i) de forma a minimizar

$$\|p^1 \star x^1 + \ldots + p^n \star x^n\|_{\max},$$

onde $u \star v$ é multiplicação coordenada a coordenada (i.e., $u \star v = (u_1 v_1, u_2 v_2, \ldots, u_m v_m)$) e $\|v\|_{\max} = \max_i v_i$.

Questão 1. Considere o seguinte algoritmo guloso: Seja $\ell_1^{t-1}, \ldots, \ell_m^{t-1}$ as cargas nas máquinas até o tempo t; o guloso atribui o job do tempo t de forma a minimizar a maior carga das máquinas após a atribuição (i.e. escolhe x^t tal que $\|\ell^{t-1} + p^t \star x^t\|_{\max}$ seja mínimo).

Mostre que pra todo $m \geq 2$ esse algoritmo guloso **não** é $(m-1)$-competitivo.

Questão 2. Considere o caso em que cada job tem o mesmo tempo de processamento em todas a máquinas (mas o jobs podem ter tempos de processamento diferentes). Mostre que nesse caso o algoritmo guloso acima é 2-competitivo.

Dica: Considere o "volume" da instancia, i.e. a soma dos tempos de processamento de todos os jobs em todas as máquinas. Qual é o minimo custo que OPT paga dado um tal volume da instancia? Baseado na análise do último job, dê uma cota inferior para o volume com relação ao custo do algoritmo. Ponha essas cotas juntas para obter o resultado.

2 Experts com garantia por intervalo

Considere o mesmo setup do problema anterior. Os algoritmos que vimos garantem número de error no **jogo todo** comparável ao do melhor expert fixo. Porém, seria interessante ter algoritmos com garantia mais robusta: **para todo intervalo** de tempo tenha, o número de erros do algoritmo nesse intervalo é comparável ao número de erros do melhor expert para esse intervalo.

Questão 1. Mostre que o algoritmo Weighted Majority (com $\varepsilon = \frac{1}{2}$, ou seja, a cada erro o peso do expert é dividido pela metade) não consegue tais garantias. Mais precisamente, considere a seguinte instância: tem-se 2 experts, e T instantes de tempo; o primeiro sempre prevê '+' e o segundo sempre '−'; a realidade é '+' até o tempo $T/2$, e '−' daí pra frente.

Mostre que nessa instância, o algoritmo Weighted Majority satisfaz:

$$\#\text{err algo na segunda metade do jogo} \geq \#\text{err melhor expert fixo para a segunda metade do jogo} + \Omega(T).$$

Questão 2. Porém, considere a seguinte variante do algoritmo Weighted Majority (vamos chamá-lo de *Fixed-Share Weighted Majority*): No tempo t, o algoritmo prevê seguindo a maioria ponderada, como no algoritmo original. Na atualização dos pesos, só atualizamos os pesos dos experts que **erraram nessa rodada** e que **tem seu peso atual w_i^t pelo menos $\frac{1}{4}$ do peso médio total dos experts**, ou seja $w_i^t \geq \frac{1}{4}\frac{W^t}{m}$, onde $W^t = \sum_{i=1}^m w_i^t$ é a soma dos pesos dos experts; para tais experts, reduzimos seus pesos pela metade. Portanto, experts que acertaram ou ja tem o peso "muito pequeno" não são atualizados.

Mostre que para todo intervalo (contiguo) I de tempos, o número de error desse algoritmo é

$$\#\text{err algo em } I \leq O(\text{OPT}_I + \log m),$$

onde OPT_I é o número de erros do melhor expert fixo no intervalo I. Para isso, siga os seguintes passos (são adaptações da prova do Weighted Majority):

1. Seja t_{init} e t_{fim} os tempos iniciais e finais de I. Considere um tempo $t \in I$ em que o algoritmo tenha errado a previsão. Seja $bad_t \subseteq [m]$ o conjunto de experts que erraram no tempo t, e seja $small_t \subseteq [m]$ o conjunto de experts que tem peso w_i^t menor do que $\frac{1}{4}\frac{W^t}{m}$. Argumente que o peso total dos experts em bad_t mas não em $small_t$ é pelo menos uma fração de W^t.

   Com isso, conclua que W^{t+1} reduz por uma fracao de W^t, e use isso para relacionar $W^{t_{fim}}$, $W^{t_{init}}$ e o número de erros sofrido pelo algoritmo no intervalo I.

2. Argumente que em todo instante de tempo e para todo expert $w_i^t \geq \frac{1}{8}\frac{W^t}{m}$.

3. Seja i^* o expert com o menor número de erros em I. Note que $w_{i^*}^{t_{fim}} \geq \frac{1}{2^{\#\text{err } i^* \text{ em } I}} \cdot w_{i^*}^{t_{init}}$.

4. Coloque essas informações juntas para obter o resultado desejado.

3 Experts com OPT dinâmico (extra)

Nesse problema você deve mostrar que no problema de previsão com experts, não temos esperança de poder compara favoravelmente contra um OPT "adaptativo" (ao invés de fixo).

Mais precisamente, relembre brevemente o setup do problema de previsão (binária) com experts: Tem-se m experts, e o jogo se desenrola em T instantes de tempo. No tempo t, recebemos a recomendação do i-ésimo expert, denotada por $e_i^t \in \{-1, +1\}$. Com a informação até o momento, o algoritmo deve decidir sua previsão $o^t \in \{-1, +1\}$. Após isso, a realidade se realiza $r^t \in \{-1, +1\}$, e o algoritmo "paga 1 unidade" caso tenha errado a previsão, ou seja, $o^t \neq r^t$.

Considere a solução ótima offline dinâmica OPT^{dyn}: munido da informação completa do jogo, essa estratégia em cada instante de tempo t seleciona um expert i suja previsão e_i^t coincida com a realidade r^t, caso tal exista. Ou seja, esse algoritmo só comete erro no tempo t caso **todos** os experts prevejam de forma errada.

Prove que para cada algoritmo determinístico[1], existe uma instancia com número constante de experts tal que

$$\#\text{ erros do algoritmo} \geq \#\text{ erros } \text{OPT}^{dyn} + \Omega(T).$$

(Em contraste, lembre que comparando com OPT fixo, que utiliza o mesmo experts in todos os instantes de tempo, o MWU obtém garantia algo $\leq \text{OPT} + O(\sqrt{T})$.)

[1] Provavelmente sua instância sera ruim mesmo para algoritmos aleatorizados, só a prova é um pouco mais tecnica; tente fazer caso possivel.

Prática: Reconhecimento de caracteres com boosting

(Exercício 10.3 ou 3.5.4, dependendo da versão, do livro "Introduction to Online Convex Optimization" do Elad Hazan.)

Baixe o training set do "MNIST database"; pode baixar em qualquer formato que encontrar, alguns examplos são:

- `http://deeplearning.net/tutorial/gettingstarted.html`
- `https://pjreddie.com/projects/mnist-in-csv/`
- `http://cis.jhu.edu/~sachin/digit/digit.html`
- `http://scikit-learn.org/stable/datasets/`
- `https://github.com/mrgloom/MNIST-dataset-in-different-formats`

Para simplificar, escolha um par de digitos para tentar reconhecer/distinguir; chamamos esse dígitos de a e b.

Primeiro, para cada um dos 28×28 pixels, crie um classificador para distinguir dígitos a e b **apenas usando esse pixel**. Depois, utilize o algoritmo de boosting baseado no Multiplicative Weights Update para combinar esse classificadores single-pixel em um classificador melhor.

Reporte brevemente a qualidade dos classificadores single-pixel e a qualidade do classificador final. Comente quantas iterações no boosting você usou. Faça um gráfico mostrando a qualidade do classificador combinado com o passar das iterações.

Comente se a hipótese da existência de weak-learner dentro dos seus classificadores single-pixel é satisfeita. Reporte também o peso que o boosting deu a cada um dos classificadores single-pixel para composição do classificador final.